

A Detailed Introduction to Parsing and Processing XML Documents with Java™ Technology – Part 2



Michael Paolini

Linux Integration Center

IBM Software Group.

paolini@us.ibm.com

Presenter:

Neil Graham

XML Parser Development

IBM Software Group.

neilg@ca.ibm.com





Topics to Cover - Part 2

- Building on the concepts from Part I.
 - f* Extra detail and examples
- Advanced topics and additional detail.
- Lots of code examples (as time permits).
 - f* Walkthrough of DOM processing code.
 - f* Walkthrough of SAX processing code.

Java's Endorsed Standards Override Mechanism

- What happens when a standard such as "org.w3c.dom" changes faster than their JRE/JDK counter parts?

f Answer: Java's Endorsed Standards Override:

- The system property `java.endorsed.dirs` specifies one or more directories that the Java runtime environment will search for such JAR files.
- Depending on the JRE version, override may be needed for `xercesImpl.jar` and `xmlParserAPIs.jar`

```
java -Djava.endorsed.dirs=\somedir\xercesImpl.jar;\somedir\xmlParserAPIs.jar
```

f The Java runtime environment will use classes in such JAR files to override the corresponding classes provided in the Java 2 Platform as shipped by Sun.

How Do I Find the Version of Xerces in Use?

f To find the version of Xerces 2 in use, use the Version class:

```
System.out.println("Parser version is " +  
    new org.apache.xerces.impl.Version().getVersion()  
    );
```

f To find the version of Xerces 1 in use, use the Version class:

```
System.out.println("Parser version is " +  
    org.apache.xerces.framework.Version.fVersionon  
    );
```





Error Handling Example

- Example source

- f* PxmlErrorHandler.java

- f* NOTE: Errors are similar in nature to exceptions.

- Xerces uses the SAXParseException as part of the error notification process.

- Demo: How XMLView handles errors.



DTD Revisited

- As we mentioned in Part 1, the DTD allows for:
 - f* Sharing your grammar/data with others.
 - f* Validation by the parser.
 - f* Defaulting of values.
 - Can query if a value was defaulted or not.
 - Care is needed as DOM can become bloated by badly designed DTDs
- Let's look a little closer at DTD defaults:



DTD Defaults

- A note to DTD writers.

- f* Bad DTDs can bloat your DOM

- f* Small XML files can cause a large DOM.

- Spotting defaulted values.

- f* You can test to see if values were defaulted by the DTD.

- f* Example 1: defaults.xml & shapes.dtd

- f* Example 2: bloat.xml & bloat.dtd



DTD Defaults (Continued)

- How can I spot defaulted values ?

f Using the ***getSpecified()*** method.

f Example: Let's look at how XMLView does this.



More about the SAX

- Home page: <http://www.saxproject.org/>
- SAX Level 1
 - f* 11 core classes and interfaces
 - f* 3 optional helper
- SAX Level 2
 - f* 19 classes and interfaces, and 10 helpers
 - f* org.xml.sax
 - f* org.xml.sax.helpers
 - f* org.xml.sax.ext

SAX2 Features and Properties

f SAX2 defines Features Flags and Properties values for the SAX parser.

- Access to these is through `org.xml.sax.XMLReader`

- ✓ `getFeature()`

- ✓ `setFeature()`

- ✓ `getProperty()`

- ✓ `setProperty()`

- A full list of features and Properties is available at:

[http://www.saxproject.org/apidoc/org/xml/sax/package-e-summary.html#package description](http://www.saxproject.org/apidoc/org/xml/sax/package-e-summary.html#package%20description)

Finding Supported SAX2 Features and Properties

f To find properties supported in SAX2:

```
//Sample from http://www.saxproject.org/?selected=get-set
try {
    String id = "http://xml.org/sax/features/validation";
    if (xmlReader.getFeature(id))
    {
        System.out.println("Parser is validating.");
    } else {
        System.out.println("Parser is not validating.");
    }
} catch (SAXNotRecognizedException e) {
    System.out.println("Can't tell.");
} catch (SAXNotSupportedException e) {
    System.out.println("Wrong time to ask.");
}
```



More about the DOM

- Home page

f <http://www.w3.org/DOM/>

- DOM Level 1 (Second Edition)

f W3C Recommendation

f <http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/>

- DOM Level 2

f W3C Recommendation

<http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/>



What Is the DOM ?

"...platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The Document Object Model provides a standard set of objects for representing HTML and XML documents, a standard model of how these objects can be combined, and a standard interface for accessing and manipulating them. Vendors can support the DOM as an interface to their proprietary data structures and APIs, and content authors can write to the standard DOM interfaces rather than product-specific APIs, thus increasing interoperability on the Web."



Or in Other Words...

- A definition of the interfaces not a description of an implementation.
- Language bindings provided for:
 - f* Java
 - f* ECMA Script
- OMG IDL also provided.
- Typically, an XML parser provides the actual implementation of these interfaces.



A More Detailed Look at DOM

- Let's review DOM in more detail...

- f* Processing (covered in XMLView walk through later)

- Documents
- Nodes
- Elements
- Attributes
- Characters

- f* Editing the DOM

- f* Saving the DOM to file (serialization).



Navigating the DOM

■ The DOM tree elements

f Can be found

- through manual transversal
- `getElementsByTagName()`
- `getElementById()`
 - ✓ The DOM implementation must have information that says which attributes are of type ID. Attributes with the name "ID" are not of type ID unless so defined.
 - ✓ ID must be unique with the document



Editing the DOM

- The DOM tree can be edited/updated

- f* Nodes can be inserted

- appendChild()
- insertBefore()

- f* Nodes can be deleted

- removeChild()

- f* Nodes can be replaced (edited)

- replaceChild()
- setNodeValue()

(Fetch the value with getNodeValue())



Editing the DOM (Continued)

- Other useful ways to edit things.

f To change the text of a TEXT node:

- `setNodeValue("New text");`

f To set the value of an Attribute for a Node:

- `setAttribute(key, value);`



DOM Editing Example (Continued)

f To remove all the children of a Node:

```
Node child;
```

```
while (child=xmlElement.getFirstChild()  
!= null)  
{  
    xmlElement.removeChild(child) ;  
}
```



DOM Editing Example

- Visual example of editing the DOM

f Example: SVGView



DOM Re-Validation

- Can an edited DOM be re-validated ?

f Not currently.

f DOM Level 2 does not support this.

f Expected in DOM Level 3.

f Therefore, no support for this in Xerces yet.



The DOM and User Data

- Useful non-standard feature.

f Provided by the Xerces NodeImpl class.

f Allows user data to be placed on a node.

- ***setUserData()***

- ***getUserData()***

f Very useful feature.

- Can be used to store a reference to a Java object.

- **Used by SVGView to gain performance.**



DOM Serialization

- You can save the DOM to a file.
 - f* Using Xerces API
 - f* Changes/Edits will also be saved

- Example 1: XMLView

- Example 2: SVGView saving edited DOM



DOM Level 2

- DOM Level 2 is now a Recommendation of the W3C.
 - f* As of November 2000
- Now split up into multiple spec parts.
 - f* Core (XML functionality)
 - f* HTML
 - f* Views
 - f* Style
 - f* Events
 - f* Traversal-Range
 - <http://www.w3.org/DOM/>
 - <http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/>

Handling Processing Instruction (PI) Nodes

- PI == Processing Instruction
- DOM
 - f* Node.PROCESSING_INSTRUCTION_NODE
- SAX
 - f* ***processingInstruction()*** method called.
- Example1: XMLView & pi.xml
- Example2: PxmlSAX2 & pi.xml



Entities

- Entities are a bit like macros.
 - f* You can define something and re-use it.
 - f* You can choose to have entities be created as DOM nodes, or have them "flattened" by the parser.
 - f* Example: XMLView & entities.xml
 - f* **NOTE:** Complex topic, refer to section 4.0 of the XML 1.0 specification.



Comment Nodes

- Comments generate special nodes.

f Node.COMMENT_NODE

f Example: XMLView and comments.xml



Handling CDATA Sections

- CDATA can appear in an XML file.
- Often used to "escape" characters that might confuse the parser.
- Special identifier is used:
f `Node.CDATA_SECTION_NODE`
- Example: `XMLView.java` & `cdata.xml`



Case Study 1 - XMLView

- Let's look at the all the source code.

- In particular, let's look at:
 - f* The DOM parsing/processing cycle.
 - f* Handling elements.
 - f* Handling attributes.
 - f* Whitespace handling
 - f* Other things worthy of note.



A More Detailed Look at SAX

- We have talked a lot about the DOM.
- Now let's look at a program that uses SAX in some more detail.
- Example 1: PxmlSAX.java
- Example 2: PxmlSAX2.java



XML Namespaces

- XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references.

```
<foo xmlns:ibm='http://ibm.com/schema'> foo </foo>
```

- The “ibm” prefix is bound to `http://ibm.com/schema` for the element “x” and it’s contents

```
<ibm:foo xmlns:ibm='http://ibm.com/schema'> bar </foo>
```

```
<ibm:foo name =“bar” />
```



XML Schema

- What do I need to change in my code?
 - f* Nothing!
- What do I gain?
 - f* Additional specification clarity.
 - Can specify types of the data.
 - f* Additional parser help.
 - Data type checking.
- What might I wish for?
 - f* DOM API wrappers to get at the type information during post parse processing of the XML.



XML Schema Example

- Updating our soccer example to use an XML Schema and more.

f Examples

- demo.xml & demo.xsd
- soccer-schema.xml & soccer.xsd

f Demo

- XMLView viewing these files.



JAXP

- Bindings for XML are being added to the Java platform via the Java Community Process (JCP)
 - f* JAXP Uses different invocation API.
 - f* Provides W3C DOM bindings.
 - f* Provides SAX API.
 - f* No XML Schema support so far
- JAXP is included in JDK 1.4
- Example: `JaxpSax.java`

Invoking the Parser using JAXP

f To invoke the use the SAX parser *via* JAXP:

```
SAXParserFactory factory = SAXParserFactory.newInstance() ;

factory.setValidating( true ) ;
factory.setNamespaceAware( true ) ;

saxparser = factory.newSAXParser() ;

saxparser.setProperty(
    "http://xml.org/sax/properties/lexical-handler", this ) ;

// fn = String & this = org.xml.sax.helpers.DefaultHandler
// in this invocation (there are 10 signitures available)
saxparser.parse( fn, this ) ;
```

Some Additional XML Technologies

■ XPath

- XPath is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer (XLink).

■ XLink

- This specification defines the XML Linking Language (XLink), which allows elements to be inserted into XML documents in order to create and describe links between resources. It uses XML syntax to create structures that can describe the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated links.

■ XQuery

- provide flexible query facilities to extract data from real and virtual documents on the Web, therefore finally providing the needed interaction between the web world and the database world. Ultimately, collections of XML files will be accessed like databases.

Some Additional XML Technologies (Continued)

■ XSL and XSLT

- XSL is a language for expressing style sheets. An XSL style sheet is, like with CSS, a file that describes how to display an XML document of a given type. XSL shares the functionality and is compatible with CSS2 (although it uses a different syntax). It also adds:
 - A transformation language for XML documents: **XSLT**. Originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language. XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data.
 - Advanced styling features, expressed by an XML document type which defines a set of elements called **Formatting Objects**, and attributes (in part borrowed from CSS2 properties and adding more complex ones.



Case Study 2 - SVGView

- A large and complex XML processor
 - f* Intensive use of XML and graphics.
 - f* Extensive use of the Java 2D API.
 - f* Performs complex XML processing.
 - f* Proof that writing the processor can be a bigger task than writing the parser.
 - f* Involves the use of many "mini-parsers"



General Reference Information

- XML in 10 points (from W3C)

<http://www.w3.org/XML/1999/XML-in-10-points>

- Books

f Many XML books are now available.

- IBM On-line references

<http://www.ibm.com/xml>

<http://www.ibm.com/java>

<http://www.ibm.com/developer>

- W3C XML Specification and news

<http://www.w3.org/XML>



Terminology Summary

- XML

 - f* Extensible Markup Language

- W3C

 - f* World Wide Web Consortium

- DOM

 - f* Document Object Model

- DTD

 - f* Document Type Definition

- SAX

 - f* Simple API for XML

Further Reading -(Specs and APIs)

- XML at W3C

<http://www.w3.org/XML/>

- SAX

<http://www.megginson.com/SAX/>

- DOM

<http://www.w3.org/DOM/>

- XML Schema

<http://www.w3.org/XML/schema.html>

Further Reading (References and Papers)

- IBM's on-line XML Tutorial.

<http://www.software.ibm.com/developer/education/xmlintro/index.html>

- IBM's XML Home Page

<http://www.ibm.com/xml>

- Xerces & Xalan

<http://xml.apache.org>

- XML.ORG

<http://www.xml.org>

- OASIS

<http://www.oasis-open.org>

- ebXML

<http://www.ebXML.org>



A Note about Today's Examples

■ Samples

f All samples are written in entirely in the Java™ programming language.

f Will be available at on the CSS CDROM

■ Runtime

f The parser being used is Xerces 2.1.0

- Available from <http://xml.apache.org>

A Detailed Introduction to Parsing and Processing XML Documents with Java™ Technology - Part 2

That's the end of the session!

Please feel free to ask more questions as time allows.

paolini@us.ibm.com